COP5536 Adv. Data Struct
Report for B+ Tree

Name: Yinuo Yang
UFID: 5725-3171
Email: yinuo.yang@ufl.edu

This project is implemented in c++ and tested by Visual Studio. The zip folder consists of three files:

1. main.cpp
2. b_plus_tree.hpp
3. b_plus_tree.cpp

This project construct three classes:

1. Pair
2. BpTree
3. BpTree_Node

Following are the detailed attributes and methods defined in above three classes:

1. Pair has two attributes: (name, type)
   key, int
   value, float

2-a. Attributes of class BpTree: (name, type, features)
   root, BpTree_Node*, a pointer to the root node
   m, unsigned int, degree of the tree

2-b. Methods of class BpTree: (name, prototype, features)
   search_key(int), Pair*, search for a key
   search_range(int, int), deque<Pair*>*, search in a key range
   insert(Pair*), void, insert a new pair into a tree
   del(int), void, delete a pair corresponding to a key
   grow(), void, increase one level of height
   shrink(), void, decrease one level of height

3-a. Attributes of class BpTree_Node: (name, prototype, features)
   m, unsigned int, same as degree of the tree
   tree, BpTree*, a pointer to the tree
   parent, BpTree_Node*, a pointer to a node's parent
   children, deque<BpTree_Node*>, a list of pointers to a node's children
   is_leaf, bool, decide if a node is a leaf
   prev, BpTree_Node*, a pointer to a node's left sibling

next, BpTree_Node*, a pointer to a node's right sibling
keys, deque<unsigned int>, a list of keys in a node
data, deque<Pair*>, a list of pointers to different pairs in a leaf node

3-b. Methods of class BpTree_Node: (name, prototype, features)
search_leaf(int), BpTree_Node*, search the relevant leaf node with a key
search_key(int), Pair*, search the relevant pair with a key
split(), BpTree_Node*, split a node into two nodes when keys overflow
insert(Pair*), void, insert a new pair into a leaf node
is_deficient(), bool, decide if a node is deficient
left_borrow(), void, borrow a pair from left sibling during delete
right_borrow(), void, borrow a pair from right sibling
left_merge(), void, merge with left sibling node when borrow cannot be used in delete
right_merge(). void, merge with right sibling node
del_root(), void, delete the root
del(int), void, delete the corresponding pair with a key
sort_keys(), void, sort the keys in an internal node
sort_children(), void, sort the children of an internal node
sort_data(), void, sort the data in a leaf node

4. Other functions: (name, prototype, features)
compare_pairs(Pair*, Pair*), bool, decide if the order of two pairs is true
compare_nodes(BpTree_Node*, BpTree_Node*), bool, decide if the order of two nodes is true